Implementing Fault Tolerance in the COTS-Based X2000 Architecture: A Case Study on IEEE 1394*

Savio N. Chau † Ann T. Tai ‡ Leon Alkalai † Huy H. Luong †

†Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

‡IA Tech, Inc.
10501 Kinnard Avenue
Los Angeles, CA 90024

Abstract

Among other challenges from NASA's X2000 Technology Development Program, affordability and miniaturization are prominent criteria, which 1) preclude the traditional solutions for mission reliability that rely on custom-built hardware and extensive component/subsystem replication, and 2) call for commercial-off-the-shelf (COTS) based approaches incorporating novel, practical fault tolerance techniques. In this paper, we report our experience in implementing fault tolerance for an IEEE 1394 compliant bus network. While IEEE 1394 adequately supports power management, high performance and scalability, its topological criteria together with the stringent power, weight and cost constraints for the X2000 spaceborne computing systems impose various restrictions on fault tolerance realization. To circumvent the difficulties, we derive a "stack-tree" topology that not only complies with the IEEE 1394 standard but also facilitates fault tolerance realization in the X2000 architecture. Moreover, by taking advantage of a unique feature of 1394 which is not originated for fault tolerance, we develop a fault-tolerant bus scheme based on an extended version of the stack-tree topology that further enhances reliability.

Keywords: IEEE 1394 bus interface, commercial-off-the-shelf, spaceborne systems, stack tree topology, fault tolerance realization

Submission Category: Practical Experience Report

Principal Contact: Ann T. Tai, a.t.tai@ieee.org

*The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

1 Introduction

NASA's X2000 is a new-generation space technology program aimed at providing an engineering model to multiple missions [1]. Currently, five missions including Pluto/Kuipai Express, Europa Orbiter, Mars Sample Return, Champollion/DS4, and Solar Probe have determined to adopt the X2000 technology. Due to the multi-mission objective, the architecture of the X2000 spaceborne computing system is driven by scalability, stringent low-cost, low-power, low-mass and low-volume criteria, yet high performance and reliability. Among other things, affordability and miniaturization (which are translated from the low-cost, low-power, low-mass and low-volume criteria) preclude the traditional solutions for mission reliability that rely on custom-built hardware and extensive component replication, calling for commercial-off-the-shelf (COTS) based approaches to high performance and fault tolerance.

In this paper, we describe how we implement fault tolerance while utilizing a COTS component, namely, IEEE 1394 bus interface [2]. This interface is chosen as the baseline for the X2000 architecture through a survey on currently available COTS interfaces and an industrial workshop held at JPL [3]. While IEEE 1394 is best qualified for X2000 (see Section 2.1 for details) with respect to most of the criteria mentioned above, it is not inherently friendly in terms of fault tolerance. In particular, the 1394 standard specifies a general tree-like bus network topology in which devices are not permitted to be connected in such a way as to form loops. From the application perspective, the stringent power, weight and cost constraints for the X2000 spaceborne systems impose further restrictions on fault tolerance realization. In particular, the use of dedicated redundancy such as spare nodes for fault-tolerant tree structures are strictly limited. Although various schemes of fault-tolerant bus network have been proposed in research literatures (see [4, 5], for example), the restrictions from 1394 and from our application prevent us from utilizing those schemes since majority of them involve either loops or spare nodes.

Based on a comprehensive study of the design implications of those constraints, we derive a simple yet flexible topology, namely, *stack-tree topology*, which not only complies with the IEEE 1394 standard but also facilitates fault tolerance realization in the X2000 architecture. In particular, by utilizing the *plug-and-play* feature of 1394 which is not originated for fault tolerance purpose, we develop a stack-tree based fault-tolerant bus scheme that permits the bus network to bypass failed MCMs (the X2000 devices implemented as *multi-chip modules* [1]) such that the surviving MCMs remain connected so long as no "adjacent failures" occur (see Section 3), enabling degradable performance and improving mission reliability. Moreover, an extended version of the stack-tree topology facilitates a fault-tolerant bus scheme to tolerate certain types of adjacent failure and thus leads to further reliability gain.

2 Selection of IEEE 1394: Benefits and Tradeoffs

2.1 Multi-Criteria Driven Bus Interface Selection

As *low cost* is the foremost important design concern for the future NASA missions, COTS-based technologies have been gaining increasing attention due to the availability and accessibility of specification, design, supporting software and test equipment. Accordingly, the search for a serial interface (which is preferred over a parallel interface for cable-harness *mass reduction*) for the X2000 architecture focuses on commercial standards, namely, 1394, Fibre Channel, SHODB and 1773, as shown in Table 1, where the survey items are associated with the X2000 design criteria. (Due to space limitation, a number of survey items are omitted here but can be found in [3].) Among the various criteria, *low power* is particularly a critical requirement for X2000 since some missions such as the Pluto/Kuipar Express travels so far from the sun that radioactive-thermal power will be the only choice for power source. However, the amount of radioactive material on-board must be limited due to spacecraft mass restriction. Since the computation requirements vary over a broad range across missions (e.g., the throughput range is from under 20 MIPS to over 100 MIPS), the X2000 architecture must be *scalable*, which in turn, requires a modularized system with standardized module interface. Together with the serial interface preference mentioned above, a loosely-coupled distributed architecture becomes the most feasible. In order to take full advantage of a loosely-coupled architecture, a multi-master bus is preferred over a command-response (e.g., the 1773 bus) because a command-response architecture will not be as scalable as multi-master due to the difficulty in load balancing. As radiation-hardened and flight qualified parts for commercial standards are not readily available, the X2000 architecture uses a *system-on-a-chip* approach, with which synthesizable ASIC cores can be procured, integrated and fabricated through radiation-hardened fabrication lines. Accordingly, the availability of synthesizable ASIC cores becomes an important requirement in the selection of serial interfaces.

Table 1 shows that 1394 is the best choice due to its power saving, scalable bandwidth and other properties satisfying X2000's design criteria. Although other COTS interfaces listed in the table have better performance than 1394, its current version (IEEE 1394-1995) has a scalable bandwidth of 100, 200 and 400 Mbps for cable implementation, which is adequate for the X2000 architecture. Moreover, IEEE 1394 has a unique feature called "plug-and-play," which refers to the following capability: Upon power-on, the bus will go through an initialization process which determines the tree configuration via node address assignment; any subsequent node addition or removal will be detected and the tree structure will be re-initialized. Nonetheless, in terms of built-in fault tolerance, only an error detection function is provided by 1394; further, the tree topology required by the standard leads to difficulties for implementing fault tolerance in a bus network, which is described below.

Table 1: Serial Interface Survey

	1394	Fiber Channel	SCI/20	1773
Power per node	1.5W (link-on), 0.75W (link-off)	3W	5W	3W
Benchmark for a possible 15 node dual bus	29.25W (with the link on/off power management)	90W	150W	90W
Bandwidth	100, 200, 400 Mbps	1 Gbps	1 Gbps	1 or 20 Gbps
Architecture	Multi master	Multi-master	Multi-master	Command response
Topology	Tree	Loop, star	Ring	Star
Arbitration protocol	Isochronous and asynchronous (fair arbitration)	Multiple (ATM, HIPPI, etc.)	ATM	N/A
ASIC core	Available	Available	Not available	Not available
Industrial support	Extensive	Extensive	Limited	Moderate
Fault tolerance	CRC error detection	CRC, Redundant ring with skip a node	CRC, Redundant ring with bypass links	Partly, Dual bus

2.2 Constraints on Fault Tolerance Implementation

Although the IEEE 1394 standard specifies the physical layer to be independently powered by the cable itself and thus can continue to transmit data even if the link layer is powered off, the physical layer and the devices attached to a bus are not ensured to function flawlessly throughout a very-long mission duration. Indeed, the physical layer which has a high voltage level and high complexity could be significantly more vulnerable to failure relative to the link layer, and a device may become faulty in a mode other than "fail-silent." Therefore, as a device and its physical layer are viewed as an integral part of a node in a bus network, to tolerate node failures is our major objective. Most network topologies inherently provide some forms of fault tolerance with respect to node failures. Examples are: A multi-drop bus can tolerate node failures so long as the failure modes do not affect the shared bus; the ring and loop topologies can tolerate a node failure by changing the direction of message transmission; an N-dimensional hypercube is guaranteed to tolerate up to $N - 1$ node failures. On the contrary, a bus network based on a tree topology is not inherently friendly in terms of fault tolerance. Specifically,

1. A single node or link failure may partition the network since there is only one path between any two nodes (no loops for fault-tolerant routing).

2. A dual bus system will not be effective in tolerating node failures if two trees have stem nodes in common (because the failure of a common stem node will partition both trees).

3. The number of hops (nodes) between any two nodes is limited to 16 by the 1394 standard, which is another restriction for fault-tolerant bus network implementation.

Besides the constraints from IEEE 1394, X2000's criteria impose further restrictions on the means for fault tolerance. In particular, the power, mass, volume and cost constraints make it

imperative to realize fault tolerance with minimal redundancy. Clearly, the constraints from IEEE 1394 combined with those from X2000 prevent us from employing those fault-tolerant bus schemes proposed in research literatures which involve loops or dedicated spare nodes, and call for a type of tree network topology that facilitates cost-effective redundancy. Further, since X2000 is intended to advance packaging technology such that all the subsystems can be put on one or two stacks of MCM, a selected topology must also support the use of the MCM-stack packaging technology. Aimed at fulfilling all those requirements, we propose a "stack tree" topology, which is described in the next section.

3 Stack-Tree Topology

In the interest of bridging the terminology between network topology and the X2000 MCM-stack packaging technology, we call the proposed topology "stack tree topology;" further, the terms "node," "device" and "MCM" are regarded as interchangeable in the remainder of this paper.

3.1 Concepts

Definition 1 *A stack tree is a tree where each stem node is connected to at most three other nodes among which at most two are stem nodes.*

For example, the trees in Figures 1(a), 1(d), 1(e) and 1(f) are stack trees (STs) while those in Figures 1(b) and 1(c) are not.

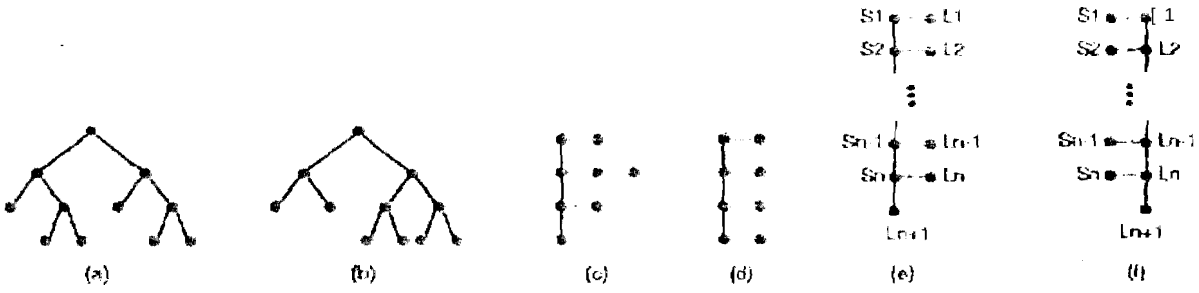

Figure 1: Trees

Definition 2 *A complete stack tree is a stack tree where each stem node is connected to at least one leaf node.*

Note that Figures 1(d) and 1(e) are complete stack trees (CSTs). Based on the CST in Figure 1(e), we can define the *CST mirror-image* as follows.

Definition 3 *The mirror-image of a complete stack tree is a tree obtained by (1) removing the edges connecting the stem nodes S_i and S_{i+1}, $i = 1 \ldots n-1$; (2) adding edges to connect the leaf nodes L_i and L_{i+1}, $i = 1 \ldots n$; (3) removing the edge connecting the nodes S_n and L_{n+1}.*

Clearly, the CST shown in Figure 1(e) and its mirror image depicted in Figure 1(f) do not have any stem nodes in common. Moreover, based on the above definitions, it can be shown that the mirror-image of a CST is also a CST.

3.2 Applications

The performance of the X2000 spaceborne systems is gracefully degradable. Accordingly, our objective is to develop fault tolerance schemes that will allow all the surviving nodes in a bus network to remain connected in the presence of failed nodes. The fact that a CST and its mirror image do not have stem nodes in common implies that losing a stem node in one tree will not partition its mirror image. Accordingly, a dual bus scheme comprising a CST and its mirror image, referred to as *CST dual scheme*, as shown in Figure 2(a) will be effective in tolerating single or multiple node failures given that the failed nodes are of the same type (all stem or all leaf) with respect to one of the CSTs. Figure 3 depicts the simplified X2000 architecture in which the CST dual scheme is implemented, where the dark and gray thick lines marked "1394 Bus" represent the primary CST-based network and the mirror image, respectively. (The thin lines marked "12C Buses" correspond to the low-speed interfaces for the low data rate engineering functions, which is out of the scope of this paper.) Note also that the stem-leaf failures clustered at the top and/or bottom of a CST (e.g., those represented by the hollow dots in Figure 2(b)), referred to as terminal clustered stem-leaf failures, will not affect the connectivity of the remainder of the tree. On the other hand, if a stem node and a leaf node in a CST dual network fail in a form other than terminal clustered stem-leaf failure (see Figure 2(c) where the hollow dots represent failed nodes), both the primary and mirror image will be partitioned.

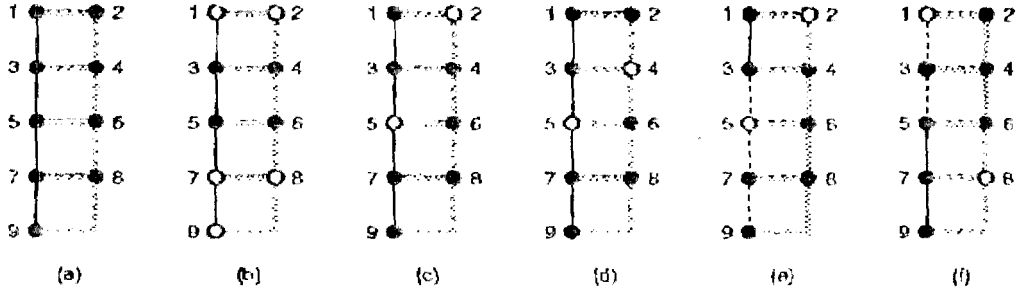

Figure 2: CST-Based Bus Network

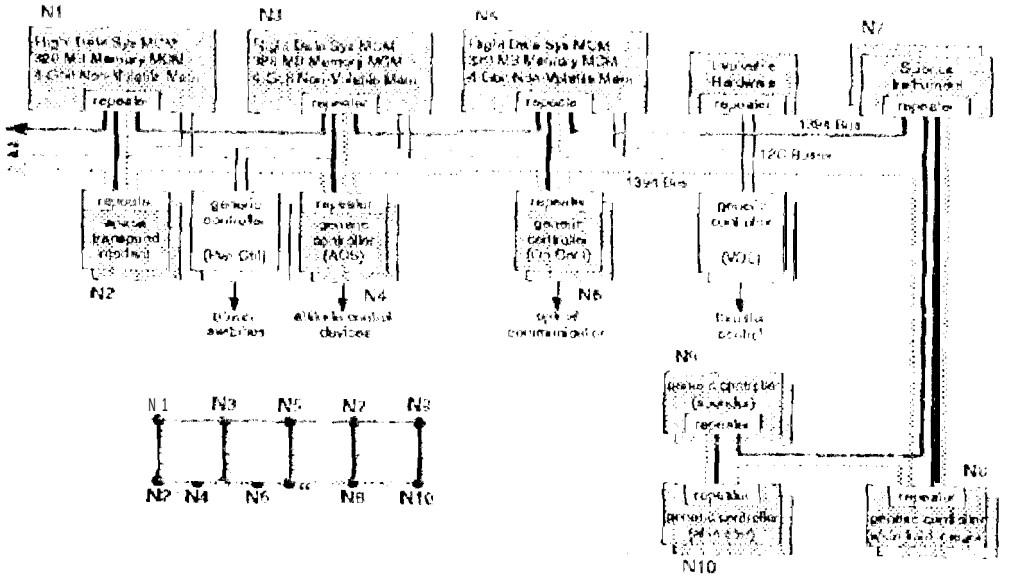

Figure 3: CST-Based Dual Bus in X2000 Architecture

To circumvent this problem, we revisit the plug-and-play feature of IEEE 1394: If a node fails, the partitioned tree will be re-initialized by the protocol and become separate but individually operational trees, even though the nodes in one of the trees may not be able to communicate with those in other trees. By taking advantage of this unique feature, a variant CST dual scheme which tolerates more types of node-failure can be implemented. Before we proceed to describe this scheme, we introduce the following terms (with respect to the CST dual scheme):

Definition 4 *A backbone edge is an edge in the primary CST or the mirror image which connects two stem nodes of the primary CST or the mirror image, respectively.*

Definition 5 *An adjacent failure corresponds to a failure scenario where a stem node in the primary CST and a stem node of the mirror image fail in a way such that the path between the two nodes contains at most one backbone edge.*

For example, the combined failure of nodes 4 and 5 in Figure 2(d) is an adjacent failure since each path between them contains only one backbone edge (edge 3-5 or 4-6). The CST-based variant dual scheme is aimed at tolerating more types of node failure by concurrently utilizing both buses to route a message. In general, two 1394 buses are not allowed to communicate directly with each other because 1) they have different address schemes, and 2) direct inter-bus communication means forming a loop. However, by implementing a detour logic in a micro-controller, a message can be routed across two bus networks. That is, a message originated from one network

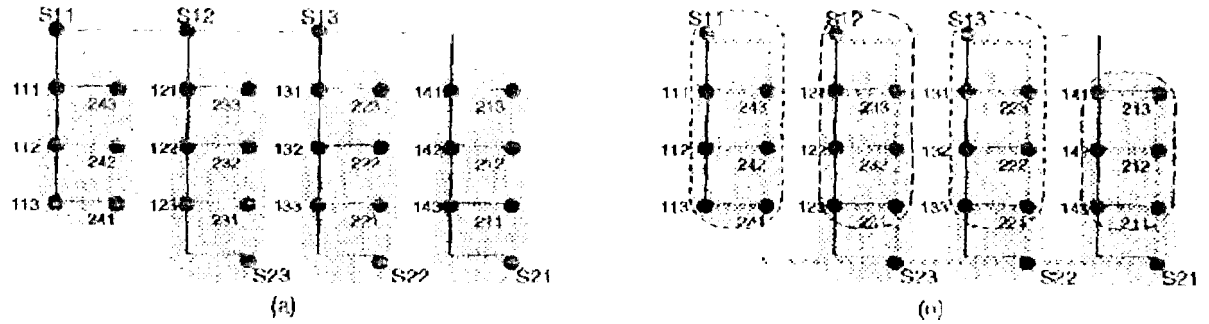

Figure 4: Hyper-CST Based Bus Network

2. A single or clustered adjacent failures (i.e., multiple adjacent failures which do not sandwich any surviving nodes) in a low-level CST which has a failed root (e.g., combined failure of nodes 121, 232, 122 and 231, plus the failure of node S12 or S23).

3. Non-clustered, multiple adjacent failures in a low-level CST (e.g., the combined failure of nodes 131, 223, 133 and 221).

4 Reliability Analysis

In accordance with the objective of the fault-tolerant bus schemes described above, we define bus network reliability is *the probability that, though a mission duration t*, the network remains in a state such that all the surviving nodes are connected. In the reliability assessment that follows, we consider only node failures because the probability of a link failure is much less significant relative to that of a node failure as explained in Section 2.2.

We begin with analyzing the CST-based bus network schemes. As explained in Section 3.2, terminal clustered stem-leaf failures in a CST will not affect the connectivity of the remainder of the tree. This observation leads us to solve the reliability of a CST-based simplex bus network (R_s^{CST}) and that of a CST-based dual bus network (R_d^{CST}) as follows. We first condition the reliability of a "remainder" by 1) its height (k) which determines the number of the possible positions of the "remainder" in the original CST, and 2) the status of the bottom leaf node (failed or operational), then uncondition it. Since the reliability of a "remainder" for a CST-based simplex network is the probability of all the stem nodes being failure-free, and that for a CST-based dual network is the complement of the probability that at least one stem node and at least one leaf node fail (in terms of the primary CST), the measures we seek to evaluate can be expressed as

$$R_s^{CST} = q\sum_{k=1}^{n}(n-k+1)(1-q)^k q^{2(n-k)} + (1-q)\sum_{k=1}^{n}(1-q)^k q^{2(n-k)}$$

Table 4: Reliability Gain from Hyper CST Based Dual Schemes ($N = 34$)

λ	R_{sys}^{HCST}	$R_{s_0}^{HCST}$	R_t^{HCST}
10^{-7}	0.86163804 32	0.9808559689	0 .99646895 74
10^{-8}	0.9852 1 83374	0.999781 5025	0.999923361
10^{-9}	0;99s51 190830	0.999997785	0.999999952 7
10^{-10}	0.99985109 11	0.9999999 778	0.999999996
10^{-11}	0.W99851081	0.99999999 8s	1.0000000000

5 Conclusion

This paper presents our experience in implementing fault tolerance for a COTS-based spaceborne system architecture, which is a big challenge today when cost concern has led to increased use of COTS products for critical applications. Since COTS products are usually not designed for critical applications, they often do not have adequate built-in fault tolerance capabilities and may not be inherently friendly in accommodating client-implemented fault tolerance functions. In implementing fault tolerance for the X2000 bus interface, we face two types of constraints: 1) the constraints from the COTS side -- the IEEE 1394 standard, and 2) the constraints from our application side -- the X2000 design philosophy. Our experience presented here suggests that a thorough understanding of the design implications of those constraints plus innovative use of the features of the COTS product in question can lead to effective solutions for fault tolerance realization.

References

[1] L. Alkalai, "NASA Center for Integrated Space Microsystems," in *Proceedings of Advanced Deep Space System Development Program Workshop on Advanced Spacecraft Technologies*, (Pasadena, CA), June 1997.

[2] I. J. Wickelgren, "The facts about FireWire," *IEEE Spectrum*, vol. 34, pp. 20–25, Apr. 1997.

[3] S. N. Chau, "X2000 avionics system conceptual design document," JPL Technical Report, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1997.

[4] C. S. Raghavendra, A. Avižienis, and M. D. Ercegovac, "Fault tolerance in binary tree architecture," *IEEE Trans. Computers*, vol. C-33, pp. 568–572, June 1984.

[5] Y.-R. Leu and S.-Y. Kuo, "A fault-tolerant tree communication scheme for hypercube systems," *IEEE Trans. Computers*, vol. C-45, pp. 641–650, June 1996.